Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4 по курсу «Моделирование»

Тема Модели на основе ДУ в ЧП с краевыми условиями 2-го и 3-го рода

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва — 2021 г.

Тема работы

Программно-алгоритмическая реализация моделей на основе дифференциальных уравнений в частных производных с краевыми условиями II и III рода.

Цель работы

Получение навыков разработки алгоритмов решения смешанной краевой задачи при реализации моделей, построенных на квазилинейном уравнении параболического типа.

Теоретические сведения

Задана математическая модель:

$$c(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}(k(T)\frac{\partial T}{\partial x}) - \frac{2}{R}\alpha(x)T + \frac{2T_0}{R}\alpha(x) \tag{1}$$

Краевые условия:

$$\begin{cases} t = 0, T(x,0) = T_0 \\ x = 0, -k(T(0))\frac{\partial T}{\partial x} = F_0 \\ x = l, -k(T(l))\frac{\partial T}{\partial x} = \alpha_N(T(l) - T_0) \end{cases} \tag{2}$$

В обозначениях уравнения лекции:

$$p(x) = \frac{2}{R}\alpha(x) \tag{3}$$

$$f(u) = f(x) = \frac{2T_0}{R}\alpha(x) \tag{4}$$

Разностная схема с разностным краевым условие при $x = 0$:

$$(\frac{h}{8}\,\widehat{c_{\frac{1}{2}}} + \frac{h}{4}\,\widehat{c_0} + \widehat{X_{\frac{1}{2}}}\,\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}} + \frac{\tau h}{4}p_0)\,\widehat{y_0} + (\frac{h}{8}\,\widehat{c_{\frac{1}{2}}} - \widehat{X_{\frac{1}{2}}}\,\frac{\tau}{h} + \frac{\tau h}{8}p_{\frac{1}{2}})\,\widehat{y_1} =$$

$$= \frac{h}{8}\,\widehat{c_{\frac{1}{2}}}\,(y_0 + y_1) + \frac{h}{4}\,\widehat{c_0}\,y_0 + \widehat{F}\,\tau + \frac{\tau h}{4}(\widehat{f_{\frac{1}{2}}} + \widehat{c_0}) \tag{5}$$

При получении разностного аналога краевого условия при $x = l$ учесть, что поток:

$$F_N = \alpha N(y_N - T_0), F_{N-\frac{1}{2}} = X_{N-\frac{1}{2}}\frac{y_{N-1} - y_N}{h} \tag{6}$$

Заданы начальные параметры:

- $k(T) = a_1(b_1 + c_1 T^{m_1})$, Вт/см K
- $c(T) = a_2 + b_2 T^{m_2} - \frac{c_2}{T^2}$, Дж/$cm^3$ K
- $a_1 = 0.0134, b_1 = 1, c_1 = 4.35 \cdot 10^{-4}, m_1 = 1$
- $a_2 = 2.049, b_2 = 0.563 \cdot 10^{-3}, c_2 = 0.528 \cdot 10^5, m_2 = 1$
- $\alpha(x) = \frac{c}{x-d}, \alpha_0 = 0.05$ Вт/cm^2 K, $\alpha_N = 0.01$ Вт/cm^2 K
- $l = 10$ см
- $T_0 = 300K$
- $R = 0.5$ см
- $F(t) = 50$ Вт/cm^2

Исходный код алгоритмов

В листинге 1 представлена реализация алгоритма решения задачи. В листингах 2 – 6 приведены вспомогательные функции и главная программа.

Листинг 1 – Реализация алгоритма решения задачи

```
1 package emission
2
3 import "math"
4
```

```
func SimpleIters() (FMat64, float64) {
    tbl := make(FArr64, int(Params.L/Params.H)+1)
    for i := 0; i < len(tbl); i++ {
        tbl[i] = Params.T0
    }
    ntbl := make(FArr64, int(Params.L/Params.H)+1)

    res := FMat64{tbl}
    ti := 0.
    fl := true

    for fl {
        ptbl := tbl
        cmax := 1.

        for cmax >= 1 {
            ntbl = getT(ptbl)
            cmax = math.Abs((tbl[0] - ntbl[0]) / ntbl[0])

            for i := 0; i < len(tbl); i++ {
                d := math.Abs((tbl[i] - ntbl[i]) / ntbl[i])
                if d > cmax {
                    cmax = d
                }
            }

            ptbl = ntbl
        }

        res = append(res, ntbl)
        ti += Params.T

        fl = false

        for i := 0; i < len(tbl); i++ {
            if math.Abs((tbl[i]-ntbl[i])/ntbl[i]) > Params.Eps {
                fl = true
            }
        }

        tbl = ntbl
    }

    return res, ti
}

func getT(tbl FArr64) FArr64 {
    lcs := getLConds(tbl)
```

```
    rcs := getRConds(tbl)

    xil := FArr64{0, -lcs.M / lcs.K}
    etal := FArr64{0, lcs.P / lcs.K}

    x := Params.H
    n := 1

    for x+Params.H < Params.L {
        tn := tbl[n]
        den := getBCf(x, tn) - getACf(tn)*xil[n]

        xil = append(xil, getDCf(tn)/den)
        etal = append(etal, (getFCf(x, tn)+getACf(tn)*etal[n])/den)

        n++
        x += Params.H
    }

    ntbl := make(FArr64, n+1)
    ntbl[n] = (rcs.P - rcs.M*etal[n]) / (rcs.K + rcs.M*xil[n])

    for i := n - 1; i > -1; i-- {
        ntbl[i] = xil[i+1]*ntbl[i+1] + etal[i+1]
    }

    return ntbl
}

func getLConds(tbl FArr64) Conds {
    var lcs Conds

    cp := getApproxPlus(getC, tbl[0], Params.T)
    kp := getApproxPlus(getK, tbl[0], Params.T)

    lcs.K = Params.H/8*cp + Params.H/4*getC(tbl[0]) + Params.T/Params.H*kp +
        Params.T*Params.H/8*getP(Params.H/2) + Params.T*Params.H/4*getP(0)
    lcs.M = Params.H/8*cp - Params.T/Params.H*kp + Params.T*Params.H/8*getP(Params.H/2)
    lcs.P = Params.H/8*cp*(tbl[0]+tbl[1]) + Params.H/4*getC(tbl[0])*tbl[0] +
        Params.F0*Params.T + Params.T*Params.H/8*(3*getF(0)+getF(Params.H))

    return lcs
}

func getRConds(tbl FArr64) Conds {
    var rcs Conds

    cm := getApproxMinus(getC, tbl[len(tbl)-1], Params.T)
```

```
101     km := getApproxMinus(getK, tbl[len(tbl)-1], Params.T)
102
103     rcs.K = Params.H/8*cm + Params.H/4*getC(tbl[len(tbl)-1]) + Params.T/Params.H*km +
104         Params.T*Params.AlphaN + Params.T*Params.H/8*getP(Params.L-Params.H/2) +
105         Params.T*Params.H/4*getP(Params.L)
106     rcs.M = Params.H/8*cm - Params.T/Params.H*km +
            Params.T*Params.H/8*getP(Params.L-Params.H/2)
107     rcs.P = Params.H/8*cm*(tbl[len(tbl)-1]+tbl[len(tbl)-2]) +
            Params.H/4*getC(tbl[len(tbl)-1])*tbl[len(tbl)-1] +
108         Params.T*Params.AlphaN*Params.T0 +
                Params.T*Params.H/4*(getF(Params.L)+getF(Params.L-Params.H/2))
109
110     return rcs
111 }
112
113 func getBCf(m, x float64) float64 {
114     return getACf(x) + getDCf(x) + Params.H*getC(x) + Params.H*Params.T*getP(m)
115 }
116
117 func getFCf(m, x float64) float64 {
118     return Params.H*Params.T*getF(m) + x*Params.H*getC(x)
119 }
120
121 func getACf(x float64) float64 {
122     return Params.T / Params.H * getApproxMinus(getK, x, Params.T)
123 }
124
125 func getDCf(x float64) float64 {
126     return Params.T / Params.H * getApproxPlus(getK, x, Params.T)
127 }
128
129 func getK(x float64) float64 {
130     return Params.A1 * (Params.B1 + Params.C1*math.Pow(x, Params.M1))
131 }
132
133 func getC(x float64) float64 {
134     return Params.A2 + Params.B2*math.Pow(x, Params.M2) - Params.C2/x/x
135 }
136
137 func getP(x float64) float64 {
138     return getAlpha(x) * 2 / Params.R
139 }
140
141 func getF(x float64) float64 {
142     return getAlpha(x) * 2 * Params.T0 / Params.R
143 }
144
145 func getAlpha(x float64) float64 {
```

```
    d := (Params.AlphaN * Params.L) / (Params.AlphaN - Params.Alpha0)
    c := -Params.Alpha0 * d
    return c / (x - d)
}

func getApproxPlus(fn func(float64) float64, n, st float64) float64 {
    return (fn(n) + fn(n+st)) / 2
}

func getApproxMinus(fn func(float64) float64, n, st float64) float64 {
    return (fn(n) + fn(n-st)) / 2
}
```

Листинг 2 – Реализация вспомогательных типов

```
package emission

// FArr64 is used to represent []float64
type FArr64 []float64

// FMat64 is used to represent [][]float64
type FMat64 []FArr64

// Conds is used to represent emission system conditions
type Conds struct {
    K float64
    M float64
    P float64
}

// Emission is used to represent emission system parameters
type Emission struct {
    A1, B1, C1, M1, A2, B2, C2, M2, Alpha0, AlphaN, L, T0, R, F0, H, T, Eps float64
}
```

Листинг 3 – Константы

```
package emission

var (
    Params = Emission{0.0134, 1, 4.35e-4, 1, 2.049, 0.563e-3, 0.528e5, 1, 0.05, 0.01, 2,
        300, 0.5, 50, 1e-3, 1, 1e-2}
)
```

Листинг 4 – Реализация функций отрисовки графика

```
package emission

import (
```

```
    "fmt"
    "image/color"
    "math/rand"
    "os"
    "time"

    "gonum.org/v1/plot"
    "gonum.org/v1/plot/plotter"
    "gonum.org/v1/plot/vg"
)

// DrawPlot is used to draw plot with given coordinates and meta info
func DrawPlot(xs, ys FMat64, title, xl, yl, file string) {
    p := plot.New()

    p.Title.Text = title
    p.X.Label.Text = xl
    p.Y.Label.Text = yl
    p.Add(plotter.NewGrid())

    for i := 0; i < len(xs); i++ {
        dots := convertDots(xs[i], ys[i])

        l, err := plotter.NewLine(dots)
        if err != nil {
            fmt.Println("Error:", err)
            os.Exit(1)
        }
        l.LineStyle.Width = vg.Points(1)
        l.LineStyle.Color = color.RGBA{
            R: uint8(genNum(0, 255)),
            G: uint8(genNum(0, 255)),
            B: uint8(genNum(0, 255)),
            A: 255,
        }

        p.Add(l)
    }

    if err := p.Save(10*vg.Inch, 4*vg.Inch, file); err != nil {
        panic(err)
    }
}

func convertDots(xs, ys FArr64) plotter.XYs {
    var conv plotter.XYs

    for i := 0; i < len(xs); i++ {
```

```
        d := plotter.XY{
            X: xs[i],
            Y: ys[i],
        }
        conv = append(conv, d)
    }

    return conv
}

func genNum(min, max int) int {
    rand.Seed(time.Now().UnixNano())
    return rand.Intn(max-min+1) + min
}
```

Листинг 5 – Реализация вспомогательных функций

```
package emission

import "math"

// Arange is used to model numpy.arange behaviour
func Arange(start, stop, step float64) []float64 {
    n := int(math.Ceil((stop - start) / step))
    rnge := make([]float64, n)
    for x := range rnge {
        rnge[x] = start + step*float64(x)
    }
    return rnge
}
```

Листинг 6 – Главная программа

```
package main

import (
    "lab_04/emission"
)

func main() {
    {
        emission.Params = emission.Emission{
            0.0134, 1, 4.35e-4, 1, 2.049, 0.563e-3, 0.528e5,
            1, 0.05, 0.01, 2, 300, 0.5, 50, 1e-3, 1, 1e-2,
        }
        res, ti := emission.SimpleIters()
        x := emission.Arange(0, emission.Params.L, emission.Params.H)
        ptsx := emission.FMat64{}
        ptsy := emission.FMat64{}
```

```
17
18          for i, v := range res {
19              if i%2 == 0 {
20                  ptsx = append(ptsx, x)
21                  ptsy = append(ptsy, v[:len(v)-1])
22              }
23          }
24          ptsx = append(ptsx, x)
25          ptsy = append(ptsy, res[len(res)-1][:len(res[0])-1])
26
27          emission.DrawPlot(ptsx, ptsy, "T(x)", "x", "T", "data/tx.png")
28
29          t := emission.Arange(0, ti, emission.Params.T)
30          s := emission.Arange(0, emission.Params.L, 0.05)
31          ptsx = emission.FMat64{}
32          ptsy = emission.FMat64{}
33          for _, v := range s {
34              r := emission.FArr64{}
35              for _, vv := range res {
36                  r = append(r, vv[int(v/emission.Params.H)])
37              }
38              ptsx = append(ptsx, t)
39              ptsy = append(ptsy, r[:len(r)-1])
40          }
41
42          emission.DrawPlot(ptsx, ptsy, "T(t)", "t", "T", "data/tx1.png")
43      }
44      {
45          emission.Params = emission.Emission{
46              0.0134, 1, 4.35e-4, 1, 2.049, 0.563e-3, 0.528e5,
47              1, 0.05, 0.01, 2, 300, 0.5, -9, 1e-3, 1, 1e-2,
48          }
49          res, ti := emission.SimpleIters()
50          x := emission.Arange(0, emission.Params.L, emission.Params.H)
51          ptsx := emission.FMat64{}
52          ptsy := emission.FMat64{}
53
54          for i, v := range res {
55              if i%2 == 0 {
56                  ptsx = append(ptsx, x)
57                  ptsy = append(ptsy, v[:len(v)-1])
58              }
59          }
60          ptsx = append(ptsx, x)
61          ptsy = append(ptsy, res[len(res)-1][:len(res[0])-1])
62
63          emission.DrawPlot(ptsx, ptsy, "T(x)", "x", "T", "data/tx2.png")
64
```

```
        t := emission.Arange(0, ti, emission.Params.T)
        s := emission.Arange(0, emission.Params.L, 0.05)
        ptsx = emission.FMat64{}
        ptsy = emission.FMat64{}
        for _, v := range s {
            r := emission.FArr64{}
            for _, vv := range res {
                r = append(r, vv[int(v/emission.Params.H)])
            }
            ptsx = append(ptsx, t)
            ptsy = append(ptsy, r[:len(r)-1])
        }

        emission.DrawPlot(ptsx, ptsy, "T(t)", "t", "T", "data/tx3.png")
    }
    {
        emission.Params = emission.Emission{
            0.0134, 1, 4.35e-4, 1, 2.049, 0.563e-3, 0.528e5,
            1, 0.05, 0.01, 2, 300, 0.5, 0, 1e-3, 1, 1e-2,
        }
        res, _ := emission.SimpleIters()
        x := emission.Arange(0, emission.Params.L, emission.Params.H)
        ptsx := emission.FMat64{}
        ptsy := emission.FMat64{}

        for i, v := range res {
            if i%2 == 0 {
                ptsx = append(ptsx, x)
                ptsy = append(ptsy, v[:len(v)-1])
            }
        }
        ptsx = append(ptsx, x)
        ptsy = append(ptsy, res[len(res)-1][:len(res[0])-1])

        emission.DrawPlot(ptsx, ptsy, "T(x)", "x", "T", "data/tx4.png")
    }
}
```

Результат работы программы

Представить разностный аналог краевого условия при $x = l$ и его краткий вывод интегро-интерполяционным методом.

Проинтегрируем уравнение на отрезке $[X_{n-\frac{1}{2}}; x_n]$, учитывая (6). Примем $F = -k(u)\frac{\partial T}{\partial x}$.

$$\int_{x_{N-\frac{1}{2}}}^{x_N} int_{t_m}^{t_{m+1}} c(t)\frac{\partial T}{\partial t}dt =$$

$$= -\int_{t_m}^{t_{m+1}} dt \int_{x_{N-\frac{1}{2}}}^{x_N} \frac{\partial F}{\partial x}dx - \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int^{t_{m+1}} t_m p(x) T dt + \int_{x_{N-\frac{1}{2}}}^{x_N} dx \int^{t_{m+1}} t_m f(x) dt \tag{7}$$

Интегрируя аналогично разностному аналогу краевого условия при $x = 0$ получим, учитывая (6):

$$\frac{h}{4}(\widehat{c_N}\ (\widehat{y_N} - y_N) - \widehat{c_{N-\frac{1}{2}}}\ (\frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} - \frac{y_N + y_{N+1}}{2})) =$$

$$= \tau(\alpha_N(\widehat{y_N} - T_0) - \widehat{X_N}\ \frac{\widehat{y_N} + \widehat{y_{N-1}}}{h}) -$$

$$- \tau\frac{h}{4}(p_N\ \widehat{y_N} - p_{n-\frac{1}{2}} - \frac{\widehat{y_N} + \widehat{y_{N-1}}}{2} + (\widehat{f_N} - \widehat{f_{N-\frac{1}{2}}})) \tag{8}$$

Приведем уравнение к виду $K_N\ \widehat{y_N} + M_N\ \widehat{y_{N-1}} = P_N$:

$$(\frac{h}{4}\ \widehat{c_N} + \frac{h}{8}\ \widehat{c_{N-\frac{1}{2}}} + \tau\alpha_N + \frac{\tau}{h}\ \widehat{X_{N-\frac{1}{2}}} + \frac{h}{4}\tau p_N + \frac{h}{8}\tau p_{N-\frac{1}{2}})\ \widehat{y_n} +$$

$$+ (\frac{h}{8}\ \widehat{c_{N-\frac{1}{2}}} - \frac{\tau}{h}\ \widehat{X_{N-\frac{1}{2}}} + \frac{h}{8}\tau p_{N-\frac{1}{2}})\ \widehat{y_{N-1}} =$$

$$= \alpha_N\tau T_0 + \frac{h}{4}\ \widehat{C_N}\ y_N + \frac{h}{8}\ \widehat{c_{N-\frac{1}{2}}}\ (y_N + y_{N-1}) + \frac{h}{4}\tau(\widehat{f_N} + \widehat{f_{N-\frac{1}{2}}}) \tag{9}$$

Используя простую аппроксимацию:

$$p_{N-\frac{1}{2}} = \frac{p_{N-1} + p_N}{2} \tag{10}$$

Получим $K_0, M_0, P_0, K_N, M_N, P_N$:

$$\begin{cases} \widehat{K_0}\widehat{y_0} + \widehat{M_0}\widehat{y_1} = \widehat{p_0} \\ \widehat{A_n}\widehat{y_{n-1}} - \widehat{B_n}\widehat{y_n} + \widehat{D_n}\widehat{y_{n+1}} = -\widehat{F_n} \\ \widehat{K_n}\widehat{y_N} + \widehat{M_{N-1}}\widehat{y_{N-1}} = \widehat{P_N} \end{cases} \tag{11}$$

Систему (11) решим методом итераций (s – номер итерации):

$$A_n^{s-1} y_{n+1}^s - B_n^{s-1} y_n^s + D_n^{s-1} y_{n-1}^s = -F_n^{s-1} \tag{12}$$

График зависимости температуры $T(x, t_m)$ от координаты x при нескольких фкисированных значених времени t_m при заданных выше параметрах.

На рисунке 1 представлен график зависимости температуры $T(x, t_m)$ от координаты x при нескольких фкисированных значених времени t_m при заданных выше параметрах.

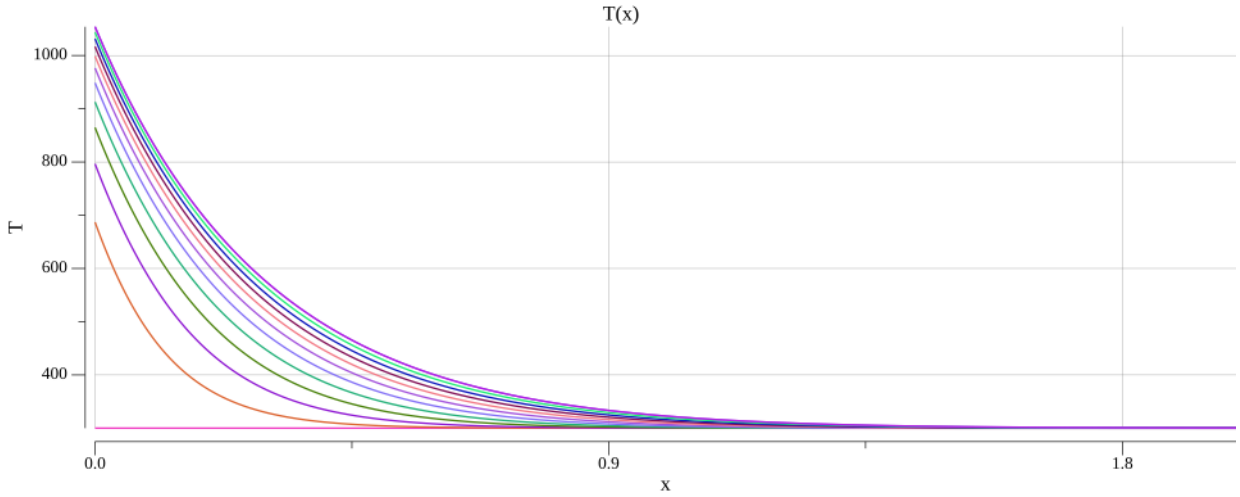

Рисунок 1 – График зависимости температуры $T(x, t_m)$ от координаты x

График зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n.

На рисунке 2 представлен график зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n. Верхний график соответствует случаю $x = 0$, нижний – $x = l$.

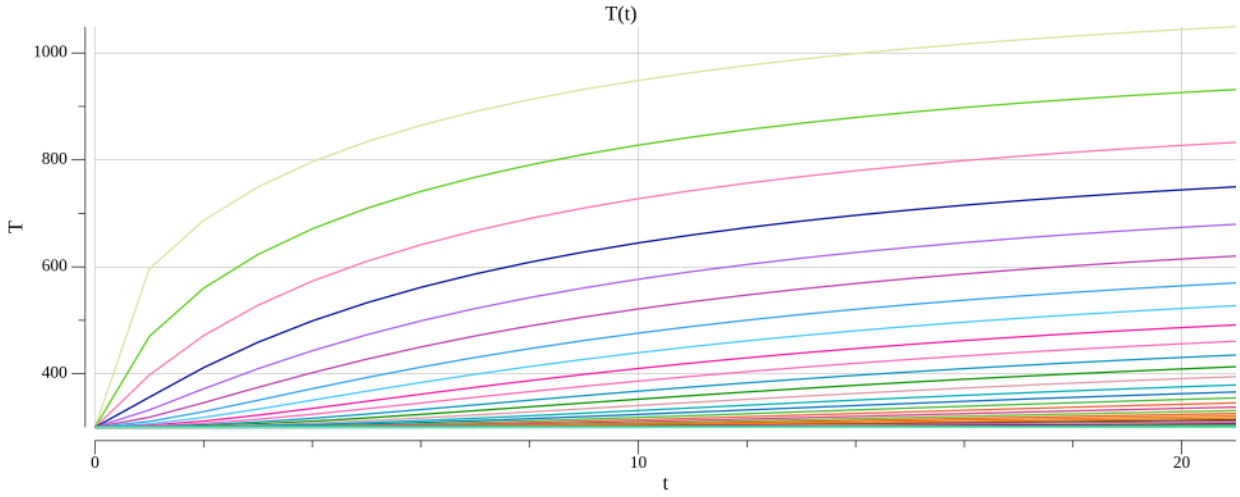

Рисунок 2 – График зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n

Приведите результаты тестирования программы (графики, общие соображения, качественный анализ).

При отрицательном тепловом потоке слева идет съем тепла (рисунки 3 – 4).

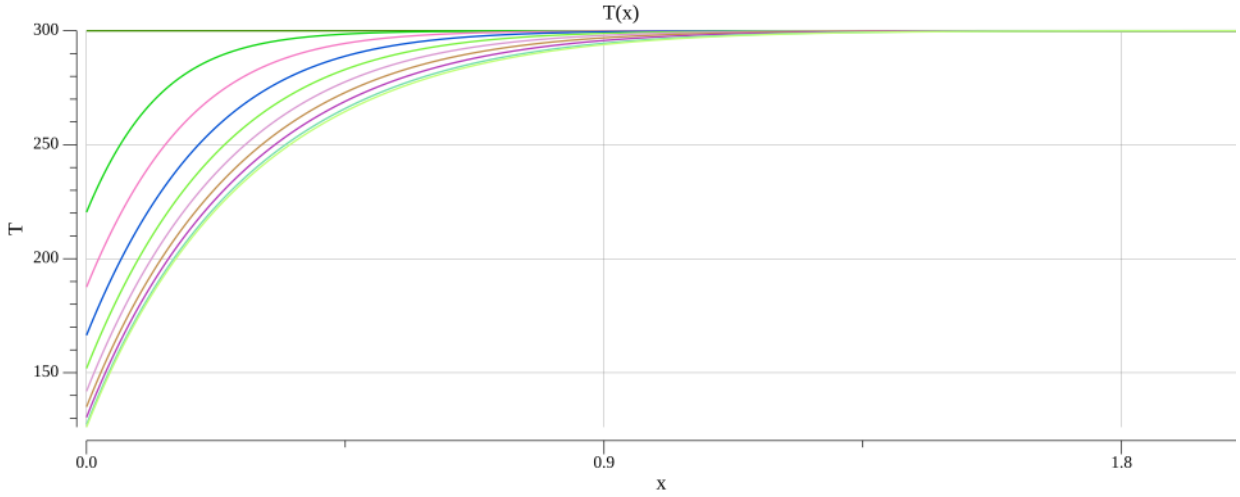

Рисунок 3 – График зависимости $T(x_n, t)$ от координаты x при $F_0 = -10$

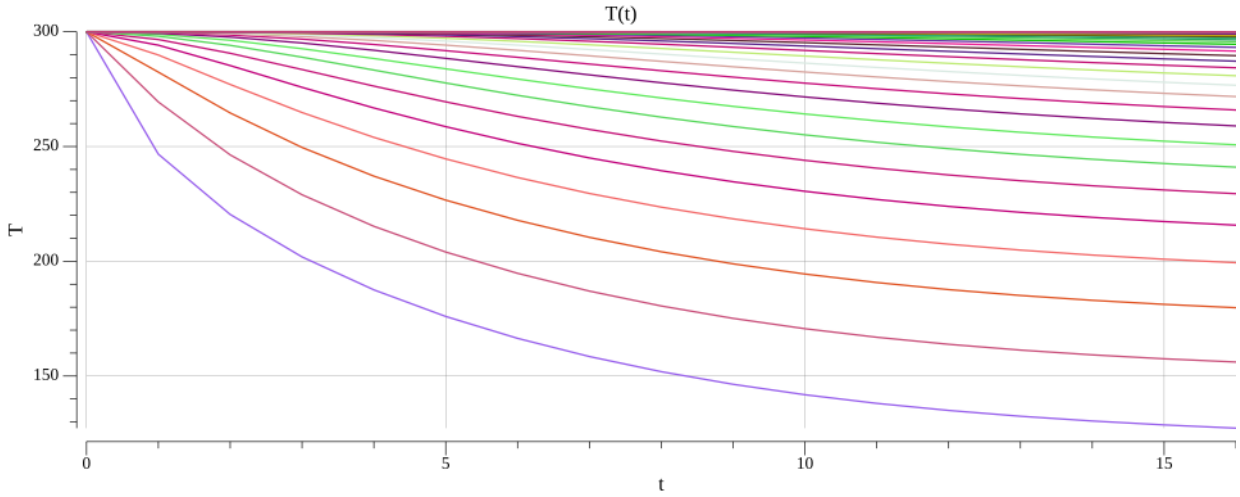

Рисунок 4 – График зависимости $T(x_n, t)$ при нескольких фиксированных значения координаты x_n и $F_0 = -10$

Если обнулить поток $F_0(T)$, то на выходе получим график температуры, установившейся в соответствии с температурой окружающей среды (рисунок 5).

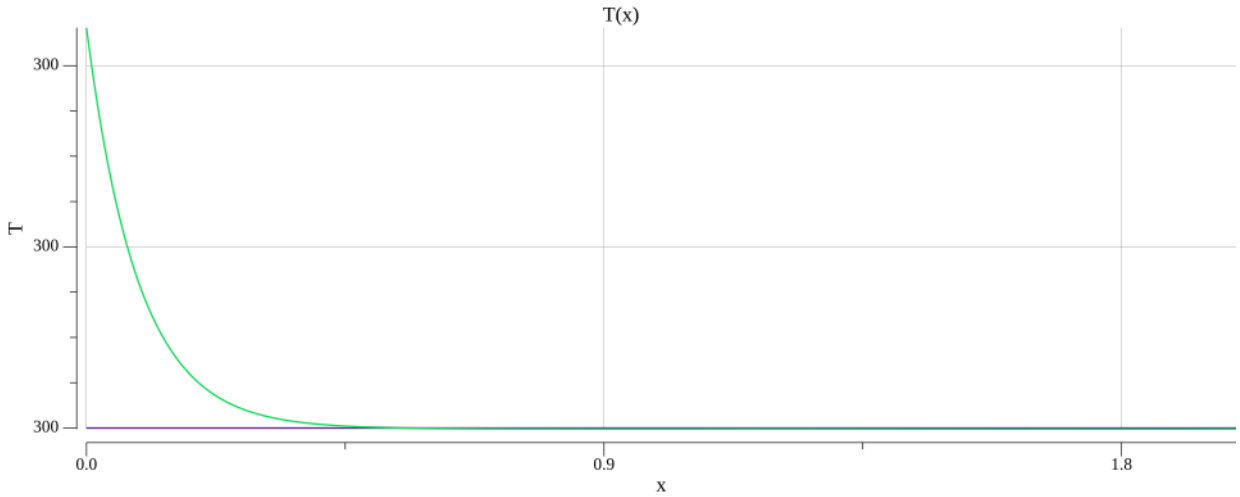

Рисунок 5 – График зависимости $T(x_n, t)$ при нулевом тепловом потоке